Table of Contents

- Table of Contents
- Milestone 1 | Tasks
  - Tasks
- Milestone 1 | Feedback
  - Problem Definition | 10 points
  - Roadmap | 10 points
  - UI | 10 Points
  - Administrative Tasks | 5 points
  - Technical Tasks | 5 points
  - Grade
- Milestone 2 | Tasks
  - Product and Project Manager | 20 Points
  - Data Scientist and Data Analyst | 20 Points
  - Database Developer | 30 Points
  - API Developer | 30 Points
  - Frontend Developer | 20 Points
- Milestone 2 | Feedback
  - Grade
- Milestone 3 | Tasks
- Milestone 3 | Feedback
  - Grade
- Milestone 4 | Tasks
- Milestone 4 | Feedback
  - Feedback
  - Grade
- Demo
- Final Grade

Milestone 1 | Tasks

Tasks

1. Problem Definition (you can learn more about it here)
2. Finalizing roles here
3. Schedule a call/meeting with me and Garo.
4. Create a product roadmap and prioritize functionality items.
5. Create a GitHub repository including `readme.md` and `.gitignore` (for Python) files.
6. Create a virtual environment in the above repo and generate `requirements.txt` (ensure `venv` is ignored in git):
    - Create venv: `python -m venv venv`
    - Activate: `source venv/bin/activate`
    - Install: `fastapi`
    - Create `requirements.txt`: `pip freeze > requirements.txt`
    - Deactivate: `deactivate`
7. Push *Problem Definition, GitHub repo setup (readme.md and .gitignore), requirements.txt*.
8. Prototype the UI using *Figma* or another similar tool.
9. Create a private Slack channel in our Workspace and name it **Group {number}**.
10. Install VS Code (also install the Project Manager extension).

Milestone 1 | Feedback

Problem Definition | 10 points

The problem is defined correctly, and the structure is well-kept.

- Broad Area of Interest.
- Preliminary Research:
  - Current trends.
  - Opportunities.
- Solution with Methodology:
  - Data Collection.
  - Analytical Techniques.
  - Implementation Plan.
- Expected Outcomes.
- Evaluation Metrics.

Grade: 10/10

Roadmap | 10 points

The roadmap seems realistic. **Perfect.**

Grade: 10/10

UI | 10 Points

Perfect!

Grade: 10/10

Administrative Tasks | 5 points

- Roles are assigned.
- Preliminary discussion with me was completed.
- Slack channel is created.
- GitHub Repo is created.

Grade: 5/5

Technical Tasks | 5 points

- Proper `.gitignore` file is available for Python.
- The `requirements.txt` file is available with pre-installed packages, indicating that `venv` was created.

Grade: 5/5

Grade

Final Grade: 40/40 - Great Job!

Milestone 2 | Tasks

Product and Project Manager | 20 Points

1. Install `mkdocs` package to start with the documentation (PSS will be available).
2. **Database schema:** Provide your product database structure (ERD).
3. Transform your project file structure according to the tree provided below.
4. Check all the below activities from your team and merge everything.

```
PythonPackageProject/ # GitHub repo
├── yourapplications/
│   ├── docker-compose.yaml
│   └── .env
│   └── service1/ # PostgreSQL
│       ├── .py files # if needed
│       └── Dockerfile # if needed
│   └── service2/ # pgAdmin
│       ├── .py files # if needed
│       └── Dockerfile # if needed
│    └── service3/ # ETL related
│       ├── .py files
│       └── requirements.txt
│       └── Dockerfile # if needed
|── example.ipynb # Showing how it works
|── docs/ # Folder needed for documentation
|── .gitignore
├── README.md
├── LICENSE
```

Data Scientist and Data Analyst | 20 Points

1. Create a new `git branch` and name it `ds`.
2. Simulate the data if needed:
    - Generate sample datasets that mimic real-world use cases.
    - Ensure data is formatted consistently for the modeling phase.
3. Use the CRUD functionality provided by the DB Developer:
    - Test the database connection and retrieve required data.
    - Implement basic CRUD operations as part of the workflow.
4. Work on the modeling part using simple models:
    - Develop basic regression or classification models.
    - Conduct feature engineering and exploratory data analysis (EDA).
    - Document initial findings and model evaluation results.
5. Push your work to the respective branch:
    - Commit codes, notes, and related documentation files.
6. Create a pull request for the Product Manager:
    - Provide a clear description of your work and any questions or suggestions.

Database Developer | 30 Points

1. Create a new `git branch` and name it `db`.
2. Create a database and respective tables as suggested by the Product Manager:
    - Build the structure using SQL scripts or Python ORM (e.g., SQLAlchemy).
    - Ensure the tables align with the ERD provided by the Product Manager.
3. Connect to SQL with Python:
    - Test the database connection using Python libraries such as `psycopg2` or `SQLAlchemy`.
    - Write code for CRUD functionality.
4. Push data from flat files to the database:
    - Load CSV or JSON files into the database.
    - Validate data integrity and consistency during the import process.
5. Add extra methods that might be needed throughout the project:
    - Write custom SQL queries or ORM methods to facilitate the API Developer's work.
    - Ensure queries are efficient and well-documented.
6. Push your work to the respective branch:
    - Include SQL scripts, connection code, and example queries.
7. Create a pull request for the Product Manager:
    - Include a detailed description of the database structure and any issues faced.

API Developer | 30 Points

1. Create a new `git branch` and name it `back`.
2. Create a new service and name it `back`:
    - Set up a FastAPI-based backend service.
    - Ensure proper directory structure and file organization.
3. Communicate with the DB Developer and PM to design the API:
    - Collaborate to define endpoints that align with database functionality.
    - Ensure endpoint naming and parameters are intuitive and consistent.
4. Create dummy endpoints initially:
    - Mock API responses to simulate real functionality.
    - Provide example JSON responses for testing.
5. Required endpoints:
    - `GET`: Retrieve data from the database (e.g., `/items`, `/details`).
    - `POST`: Insert new data into the database.
    - `PUT`: Update existing records in the database.
    - `DELETE`: Remove data from the database.
6. Push your work to the respective branch:
    - Include fully functional endpoints with initial tests and example requests.
7. Create a pull request for the Product Manager:
    - Document endpoint functionality, parameters, and example usage.

Frontend Developer | 20 Points

1. Create a new `git branch` and name it `front`.
2. Create a container/service and name it `front`:

    - Use a lightweight web server like Nginx to host the frontend.
    - Ensure the Dockerfile and `docker-compose.yaml` are updated accordingly.
3. Collaborate with the PM to create the skeleton of the website:
    - Develop basic page layouts and structure using HTML/CSS.
    - Include placeholders for API integration and dynamic elements.
4. Push your work to the respective branch:
    - Ensure all files are neatly organized with clear documentation.
5. Create a pull request for the Product Manager:
    - Highlight completed features and areas needing review or discussion.

Milestone 2 | Feedback

Grade

Reviewed and increased by 30 points based on your performance !

Final Grade: 60/120

Milestone 3 | Tasks

Were provided via **Slack.**

Milestone 3 | Feedback

Were provided via **Slack.**

Grade

Grade: 84/120

Milestone 4 | Tasks

Were provided via **Slack.**

Milestone 4 | Feedback

Feedback

- Documentation and README updates were **great**.

Grade

Final Grade: 100/100

Demo

Excellent!

Final Grade: 20/20

Final Grade

Final Grade: 304/400